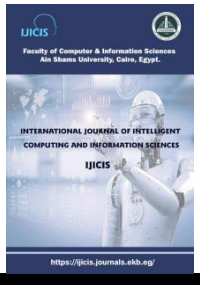
IJICIS, Vol.23, No.3, 95-113
DOI: 10.21608/ijicis.2023.217108.1279

International Journal of Intelligent Computing and Information Sciences

https://ijicis.journals.ekb.eg/



A RECOMMENDER SYSTEM FOR FAULT RECOVERY STRATEGIES IN WEB SERVICES COMPOSITION TESTING

Roaa A. ElGhondakly

Department of Information Systems, Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt
roaa.ahmed@cis.asu.edu.eg

Sherin M. Moussa*

Department of Information Systems, Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt
Laboratoire Interdisciplinaire de l'Université Française d'Egypte (UFEID Lab), Université Francaise d'Egypte, Cairo 11837, Egypt
sherinmoussa@cis.asu.edu.eg,
sherin.moussa@ufe.edu.eg

Nagwa Badr

Department of Information Systems, Faculty of Computer and Information Sciences, Ain Shams University
Cairo 11566, Egypt
nagwabadr@cis.asu.edu.eg

Received 2023-06-12; Revised 2023-06-12; Accepted 2023-07-12

Abstract: *Faults recovery has recently emerged as an important aspect of web service composition (WSC) testing, as it aims to minimize the impact of faults on system functionality through restoring the system's operation after a fault has occurred. However, most of the existing recommendation systems (RSs) tend to recommend frequently used services, which lack diversity and face inaccuracies due to incomplete or biased historical data. In addition, the focus of the existing RSs is on proposing fault handling models rather than recommending the best recovery strategy for handling faults, with most being code-based, thus not suitable for WSCs. Accordingly, due to the opaque nature of WSCs with hidden source code, model-based recovery methods are preferred. In this paper, the Fault Recovery Strategies RS for WSCs (F2RS-WSC) is proposed to recommend the best recovery strategy for handling emerging faults in the WSCs paradigm. The proposed system is a model-based system that recommends the best strategy for recovering faulty paths generated from service dependency graphs (SDGs) based on the faults` types, severity levels, faults` location, as well as the time at which faults may occur. The experimental results show that the time consumed by F2RS-WSC to recommend the optimum recovery strategy represents less than 3% of the SDG parsing time and 4% of the path validation time. In addition, its superior performance assures its accuracy and efficiency. Thus, it achieves accuracy levels between 70% and 88 % among multiple datasets. Moreover, its average precision, recall and f-measure values are 0.85,0.81 and 0.86 respectively.*

Keywords: *Fault recovery strategies, fault handling, service-oriented computing, web service composition, service dependency graph, recommender system, model-based applications.*

*__Corresponding Author__: Sherin M. Moussa

Information Systems Department, Faculty of Computer and Information Science, Ain Shams University, Cairo 11566, Egypt

Laboratoire Interdisciplinaire de l'Université Française d'Egypte (UFEID Lab), Université Francaise d'Egypte, Cairo 11837, Egypt

Email address: *sherinmoussa@cis.asu.edu.eg, sherin.moussa@ufe.edu.eg*

1. Introduction

Due to the rapid expansion of various distributed computing models, service-oriented computing (SOC) has become increasingly intricate and has gained recognition as an emerging trend. SOC models are constructed by assembling a collection of loosely-coupled distributed services, forming web service compositions [1,2]. These SOC models encompass software reuse, striving to sustain superior levels of performance, accuracy, and quality. However, since they are built upon pre-existing services, these systems encounter various challenges. These challenges include interdependencies among services [2], complexities in integration [3], and ensuring quality of service (Q oS) metrics such as response time, throughput, and availability [2,4]. Integration and dependency challenges are prominent concerns in SOC paradigms and web service composition systems [2,5], which can lead to performance degradation [6] and the occurrence of faults [1,7]. Predicting faulty components and facilitating fault recovery in web services testing has become increasingly challenging. Web services testing is primarily conducted as black box testing, as the internal code of web services is concealed, and only input and output parameters are accessible [8]. Consequently, leveraging model-based testing proves to be more effective and reliable, particularly for testing web service compositions. Therefore, the development of model-based testing approaches for testing the integration and dependencies of web service compositions becomes crucial. In this regard, employing service dependency graphs (SDGs) for testing web service compositions offers an efficient and reliable approach. SDGs are based on capturing the dependencies between services, rather than solely representing the services in a graphical form, distinguishing it from other types of graphs [2,5].

SOC testing has been conducted from various perspectives, including fault tolerance encompassing fault detection, recovery, prediction, injection, and localization [1]. However, this study will primarily focus on fault tolerance and recovery. Effective and successful fault recovery methods in the early stages significantly enhance the reliability, availability, and quality of the System Under Test (SUT) [9,10]. When faults are present in the SUT, the system may either terminate execution or behave differently than expected [9,10]. Consequently, fault recovery methods aim to assist the system in overcoming fault occurrences by either terminating the execution process or aborting and leaving the system in a safe state [9,10]. Fault recovery encompasses various strategies for addressing errors, including retry, substitution, compensation, roll-back, replication, and checkpointing [9,10]. Among these strategies, retry, replication, substitution, and checkpointing lead to the termination of the execution process, while roll-back and compensation strategies allow the execution to proceed by aborting the faulty service and ensuring the system's safety [10–13]. In the retry strategy, the system attempts to invoke the faulty service again, but if the fault persists, it switches to another recovery strategy [9,10]. Substitution involves replacing the faulty service with a non-faulty one, similar to compensation [14–16]. Replication entails replacing the faulty service with a backup replica once a fault occurs [11] . Checkpointing involves creating checkpoints after each system change, allowing a task to restart from the most recent checkpoint instead of the beginning in case of failure [12,13]. In the checkpointing strategy, the faulty service is substituted with another service that adheres to quality of service (QoS) constraints [12,17].

However, the majority of current recommendation systems tend to suggest commonly used services for recovering faulty services in a given composition, resulting in a lack of diversity [18–20]. These systems also encounter inaccuracies caused by incomplete or biased historical data [18,19]. Moreover,

existing recommender systems primarily concentrate on suggesting fault handling models instead of recommending the optimal recovery strategy for fault management [19,20]. Most of these systems are code-based, making them unsuitable for web service composition [18–20]. Consequently, given the opaque nature of web service compositions that involve hidden source code, model-based recovery methods are preferred.

This paper introduces the Fault Recovery Strategies Recommender System for web service compositions (F2RS-WSC), which aims to recommend the optimal recovery strategy for addressing emerging faults in the web service compositions paradigm. The proposed system operates based on a model-based approach and provides recommendations for recovering faulty paths derived from service dependency graphs (SDGs), to maintain dependencies between services, thereby enhancing the reliability and efficiency of the composition [2,5], taking into account factors such as fault types, severity levels, fault locations, and the timing of potential faults. The key contributions of this study can be summarized as follows:

1. Introducing the Fault Recovery Strategies Recommender System for web service compositions (F2RS-WSC) approach designed specifically for testing model-based web service compositions. This innovative approach effectively resolves the common problem of limited access to web service source code, a concern frequently encountered in similar research studies.
2. F2RS-WSC aims to recommend the optimal recovery strategy for addressing emerging faults in the web service compositions paradigm considering factors such as fault types, severity levels, fault locations, and the timing of potential faults. Instead of choosing random strategies that might not suits the type of error, severity level, fault location …etc.
3. F2RS-WSC utilizes Service Dependency Graphs (SDGs) to operate, ensuring the retention of integration and dependency relationships among services within a web service composition. This sets it apart from conventional model-based studies that often employ more abstract connections, lacking the specificity and granularity provided by SDGs.

2. Related Works

Fault recovery plays a crucial role in testing web service composition, enabling the creation of a more resilient and smoothly functioning system that can promptly recover from encountered faults [1,21] This, in turn, enhances availability, scalability, and composition quality, while minimizing costs and efforts under expected operational conditions [22,23]. Numerous studies have addressed fault tolerance and recovery strategies for Service-Oriented Computing (SOC) systems [1,13,14,21,22,24,25]. However, the majority of these studies focus on proposing recovery approaches for substituting the faulty service rather than considering any other recovery strategy. Thus, in [15] authors proposed a fault tolerance approach which is code-based not model-based for web service compositions through solely recovering the faulty service and performing QoS ranking techniques for selecting the best service suitable for recovery. However, Code availability in SOC paradigms is a major challenge. In addition, the service recovery strategy lacks integration assurance. In [16], the recovery approach applies Single service (SSR) and multiple service reconfiguration (MSR) to select a service substitution while searching for a nearby solution, in addition to Harmony Search (HS) algorithm to speed up the recovery process. However, it only considers substitution strategy and ignores other recovery strategies.

Another code-based recovery approach was proposed in [11] by applying ranking algorithms as PageRank based Service Component Ranking (PSCR) and HITS based Service Component Ranking (HSCR) Algorithms. Yet, it was a code-based approach which encounters the nature of web service concerning code availability. While in [9] the authors adopt the retry strategy through proposing self-healing approach based on causes of faults. However, it is a code-based approach that contradicts the nature of web services concerning source code availability. In [14], a model-based approach was proposed for examining fault tolerance in web service composition. It adopted petri nets as a formal model to model services and study the interactions between them for achieving optimum fault tolerance model. It is a 3-phase utilizing QoS-aware parameters of services through invocation, synchronization, and exception. When the resources of services increase, the proposed approach's reliability is affected due to the incompetence of this mechanism. In [17] the authors proposed a recovery approach based on checkpoint rollback strategies. Thus, it provides dynamic load balancing for cloud computing using ant colony optimization algorithm resulting in minimizing search time and boosting performance of network. However, the results from the model simulations are not presented. Other studies adopted checkpointing strategy as in [12,13,17] while some approaches were code-based however web services nature doesn`t guarantee source code availability. However, some recommender systems were proposed as [26,27] for fault recovery in web service composition. However, in [27] the authors proposed a recommender system that apply substitution strategy only for substituting the faulty service through selecting another one based on their location in server and some QoS attributes. While, in [26] the proposed recommender system is able to recommend the best recovery strategy among 3 fixed strategies according to the execution time. However, most of related studies and recommender systems focus only on one recovery strategy, which is substitution strategy. In addition, they consider QoS parameters in selecting the substituting service. Meanwhile, most of them ignore important factors for choosing the optimum recovery strategy or method as type of occurred fault, its severity level, its location, and time at which fault occurs. Moreover, the proposed recovery approaches were code-based which comply with the nature of web service as code availability is not guaranteed which will affect the performance and accuracy of their approaches.

The adopted recovery strategies in (F2RS-WSC) approach are retry, substitution, roll-back, replication, and checkpointing [11,16]. An accompanying comprehensive depiction of each strategy, along with prior relevant research exploring it as follows:

1. Retry : The system attempts to execute the faulty service again, but if the fault persists, it switches to another recovery strategy [9,10].
2. Substitution : This strategy tend to replace the faulty service or faulty composition with a non-faulty one [14–16].
3. Roll-back: in this strategy, the system will roll-back to the last stable state before fault occurs [17].
4. Replication : in this strategy, a backup replica will replace the faulty service once a fault occurs [11].
5. Checkpointing: it involves creating checkpoints after each system change, allowing a task to restart from the most recent checkpoint instead of the beginning in case of failure [12,13,17].

As shown in Table 1, a comparison between multiple fault tolerance approaches in terms of adopted recovery strategy, whether it is code-based or model-based and the criteria under which the recovery

strategy is selected. From these criteria, type of fault, level of severity, and the timing of the fault occurrence, location of fault and QoS parameters. Despite the existence of multiple related studies in fault recovery and fault tolerance paradigm, most of these studies adopt substitution strategy by replacing the faulty service with a non-faulty one. In addition, they neglect important criteria as type of fault, level of severity, and the timing of the fault occurrence and location of fault while choosing the recovery strategy [14,27]. Moreover, The primary focus was on source code metrics, but their use is limited to specific programming languages and not suitable for opaque nature of services in SOC systems [12,14,26,27]. Accordingly, a model-based recovery strategy recommender system is needed to detect the optimum recovery strategy based on fault aspects rather than applying random recovery strategy.

Table 1 Summarized Comparison between Proposed F2RS-WSC and Related Studies

Ref#	Code-Based/ Model-Based	Recovery Strategy	Strategy Selection Criteria	Recommender System
[15]	Code-Based	Substitution	QoS parameters	No
[16]	Code-Based	Substitution	-	No
[11]	Code-Based	Replication	-	No
[9]	Code-Based	Retry	Causes of Faults	No
[14]	Model-Based	Substitution	QoS parameters	No
[17]	Code-Based	Roll-Back and Checkpointing	-	No
[12]	Code-Based	Checkpointing	-	No
[27]	Model-Based	Substitution	Fault Location and QoS Parameters	Yes
[26]	Code-Based	Roll-back, Retry, Substitution and Replication	Execution state when a failure occurs, context-information, QoS Parameters	Yes
F2RS-WSC	Model-Based	Retry, Substitution, Replication, Roll-back and checkpointing	Fault Type, Fault Severity Level Fault Location, Time at which fault occurs	Yes

3. The Proposed Fault Recovery Strategies Recommender System for web service compositions (F2RS-WSC) Approach:

In this section, the proposed Fault Recovery Strategies Recommender System for web service compositions (F2RS-WSC) approach was introduced. The F2RS-WSC approach is a model-based recovery recommender system. Thus, The model-based techniques convey web services nature with hidden source code [1–3,28]. The F2RS-WSC approach aims to recommend the optimal recovery strategy for addressing emerging faults in the web service compositions paradigm. The proposed system operates based on a model-based approach and provides recommendations for recovering faulty paths derived from service dependency graphs (SDGs), to maintain dependencies between services, thereby enhancing the reliability and efficiency of the composition [2,5,28], taking into account faults multiple

aspects of faults such as fault types, severity levels, fault locations, and the timing of potential faults. Figure 1 represents the system Architecture of the proposed F2RS-WSC approach.

As shown in Figure 1, the proposed F2RS-WSC architecture consists of three main modules: Web Service Composition Preprocessing, Faulty Path Detection and FRS-WSC Recommendation. It accepts SDGs as an input. It examines the web service composition to determine whether it is faulty or not. It also specifies different aspects of faults such as fault types, severity levels, fault locations, and the timing of potential faults. Hence, for faulty edges, a recovery strategy is recommended based on the detected faults` aspects.

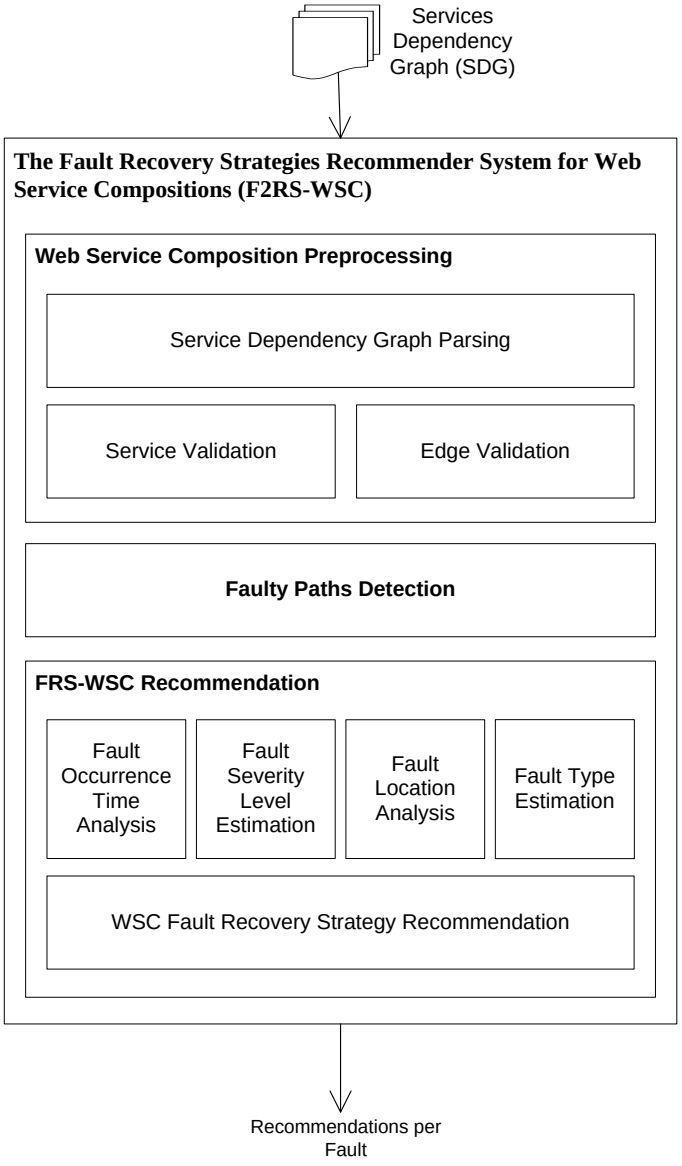


Figure. 1: The Proposed F2RS-WSC Approach System Architecture

For a comprehensive explanation of the F2RS-WSC architecture, an illustrative example is deployed that adopts the service repository in [2] and its corresponding SDG generated by the Mutual

Information-based Services Dependency (MISD) model in [5]. Figure 2 represents the generated SDG by the MISD model [5]. This SDG differ than any SDG as it is constructed based on the Web Service Mutual Information (WSMI) value between services in addition to other criteria that guarantee an accurate dependency level between each two services rather than parameters matching as in other SDGs in [2,28,29]. As represented by Figure 2, the SDG is a layered graph consisting of 9 services with their corresponding input/output parameters. Each layer contains a set of services that have input parameters matching the output parameters of services in the previous layers, while their output parameters match the input parameters of the services in the following layer. In addition, the weights on the edges represent the WSMI value calculated by the MISD model in [5]. In the following sub-sections, a detailed description for the three main modules is presented.

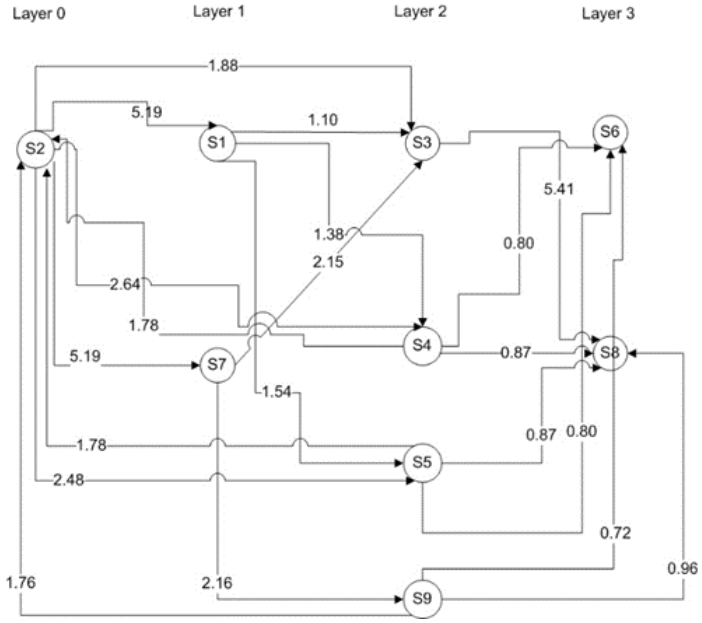


Figure. 2: The illustrative SDG example used by MISD model in [5]

3.1.Web Service Composition Preprocessing

This module acts as a foundation stage for the F2RS-WSC approach, in which it receives the SDG as an input and validates the given web service composition. The following sub-sections illustrate how this module works.

3.1.1. Service Dependency Graph Parsing:

In the F2RS-WSC approach any SDG can be used as an input. This sub-module is responsible for parsing the SDG, and all its data, as nodes, edges, and layers, are stored in a database table. Following our illustrative example in Figure 2, the SDG constructed by the MISD model in [5] is given as an input to the F2RS-WSC approach.

3.1.2. Service Validation:

This sub-module is responsible for validating each service in the web service composition path under test. F2RS-WSC splits the composition under test into edges and searches the SDG for source and target

services, forming the edges to confirm their existence. Hence, if at least one service does not exist in the SDG, the whole composition is considered faulty. Following the illustrative example in Figure 2, assume $\boldsymbol{path\ 1}: \boldsymbol{S2, S1, S3, S8}$ is given to F2RS-WSC to validate this composition. First, the F2RS-WSC splits it into 3 edges: $\boldsymbol{E1}$: "$\boldsymbol{S2, S1}$", $\boldsymbol{E2}$: "$\boldsymbol{S1, S3}$" $\boldsymbol{and\ E3}$: "$\boldsymbol{S3, S8}$". It then searches the SDG looking for all services in each edge. Since all services "$\boldsymbol{S2}$", "$\boldsymbol{S1}$", "$\boldsymbol{S3}$ " $\boldsymbol{and}$ "$\boldsymbol{S8}$" exist in the graph, F2RS-WSC continues to the next step "Edge Validation". However, to validate $\boldsymbol{path\ 2}: \boldsymbol{S2, X, S3}$, F2RS-WSC splits it into 2 edges; $\boldsymbol{E1}$: "$\boldsymbol{S2, X}$" $\boldsymbol{and\ E2}$: "$\boldsymbol{X, S3}$", in which service "$\boldsymbol{X}$" in $\boldsymbol{E1}$ does not exist in the SDG. Therefore, the validation process terminates immediately, declaring that this composition is faulty and moves to "FRS-WSC Recommendation" to start recommending the optimum recovery strategy for every faulty edge based on fault types, severity level…etc.

3.1.3. Edge Validation :

In this sub-module, the web service composition under test is split into edges. Each edge is examined whether it is faulty or non-faulty. Thus, F2RS-WSC validates separate services through validating the source and target nodes and confirms their existence in the SDG as well validating the edge connecting these two services. However, the source and target service validation do not guarantee the validation of edges between them. Thus, both nodes might exist in the SDG under test with no edge between them, which makes this edge faulty. Thus, following our illustrative example, assume providing F2RS-WSC approach $\boldsymbol{path\ 1\ as}: \boldsymbol{S2, S1, S3, S8}$ for validation. F2RS-WSC splits its edges and validates each edge separately. So, the edge between "$\boldsymbol{S2, S1}$" is checked; if it exists, the edge is then valid and marked as non-faulty edge; otherwise, if it does not exist or one of its services` nodes, it is marked as a faulty edge. Hence, for path1, all edges are non-faulty. Therefore, F2RS-WSC declares path 1 as non-faulty. On the other hand, given $\boldsymbol{path\ 2}: \boldsymbol{S2, X, S3}$, the edge between "$\boldsymbol{S2, X}$" is declared as a faulty edge, because node "$\boldsymbol{X}$" does not exist in the SDG as well as the edge.

3.2.Faulty Path Detection

In this module, a faulty path is detected after validating all services and edges forming this path. Hence, after performing service validation and edge validation processes. The F2RS-WSC will declare if the path is faulty or non-faulty based on existence of any faults either on services level or edges level. Thus, following our illustrative example, given the two paths $\boldsymbol{path\ 1\ as}: \boldsymbol{S2, S1, S3, S8}$ and $\boldsymbol{path\ 2}: \boldsymbol{S2, X, S3}$ the F2RS-WSC declares path 2 as faulty and considers a recommendation for a recovery strategy for it in the following sub-modules.

3.3.FRS-WSC Recommendation

This module is the core of the proposed F2RS-WSC recommender system. Thus, it illustrates how the recommender system works showing the factors adopted to recommending the optimum recovery strategy. Hence, F2RS-WSC provides recommendations for the optimum recovery strategy needed to handle faults occurred per edge in web service composition. After declaring faulty composition, the F2RS-WSC approach builds its recommendations based on type of fault, level of severity, and the timing of the fault occurrence, fault's locations, and the time at which the faults occur.

3.3.1. Fault Type Estimation:

The proposed F2RS-WSC approach focuses on identifying the faults in the model-based service-oriented architectures (SOA). This is because the source code of these architectures is not available and are therefore treated as black box components[1,7]. Therefore, the proposed approach is more suitable for handling the faults caused in the SOA systems, which are based on the cause of fault [1,7]. It helps to identify the fault types and their causes, thereby making it easier for the developers to fix the issues[1,7]. The utilization of this approach can have advantages in the enhancement of model-based SOA systems' performance. In the F2RS-WSC approach, five categories of faults are taken into account, including:

1. Right-First-Time failure: which materializes when the initial service in the sequence is either unavailable or flawed.
2. Control Flow error: which arises when an endless loop or the backtracking to a wrong state or place is detected .
3. Dependency/integration fault: which is associated with inaccuracies in the connection between services .
4. Availability fault: which verifies whether a service exists in the specified repository or not.
5. Test path delay fault: which indicates any delay in the tested sequence, such as having a loop or an indirect route.

3.3.2. Fault Severity Level Estimation:

The severity level of predicted faults is determined through the proposed F2RS-WSC approach, which expresses the potential impact on the overall system. Severity levels may range from high (which can lead to system failure), to low (which have a minor impact on system functionality) [30,31]. The F2RS-WSC approach maps severity levels with various predicted fault types, including Blocker, Critical Error, Major Severity Level, Minor Error, and Low Severity Level.

1. Blockers are bugs that block further testing, causing system crashes in specific environments, which is suitable for the “Right-First-Time failures.”
2. Critical Errors represent security issues that could lead to system shutdown, data loss, or other severe damage, which is suitable for the “Control Flow error.”
3. Major severity levels negatively affect large portions of the system and manifest in certain types of testing, which is suitable for the “Dependency / integration fault.”
4. Minor Errors do not impact the basic functions of the system or testing process, which is suitable for the “Availability fault.”
5. Low Severity Levels represent bugs with limited impacts found during user interface testing, which is suitable for the “Test path delay fault”.

3.3.3. Fault Location Analysis:

The function of this sub-module is to detect faults in a test path by analyzing the Service Dependency Graph (SDG) for any presence of faulty services or edges. The process of identifying faults begins with analyzing the test path, dividing it into segments or edges consisting of two connected services, which are then examined for any issues. If a fault is identified on an edge, its location is determined by F2RS-WSC approach, indicating the specific edge where the fault exists. In summary, the fault's location pinpoints the edge that contains a faulty service.

3.3.4. Fault Occurrence Time Analysis:

This sub-module is responsible for measuring the time at which faults occur. Thus, measuring the time of fault occurrence indicates the specific moment or period when a fault, error, or failure happens in a system or process. It is a crucial metric in determining the root cause of the fault and developing effective solutions to prevent similar issues from happening in the future. In other words, it provides a timestamp for when a fault occurs, which can be used by F2RS-WSC approach to track the fault and determine the optimum recovery strategy that is eligible to handle it taking into consideration other aspects as fault type, severity level and fault location.

3.3.5. WSC Fault Recovery Strategy Recommendation:

This sub-module is the heart of the F2RS-WSC recommender system where the recommendations of optimum recovery strategies take place. Thus, the decision is made by F2RS-WSC to select the best recovery strategy based on Fault type, fault level of severity, fault location and time at which fault occurs. Hence, a detailed demonstration of all these aspects is given in the previous sub-section. Accordingly, F2RS-WSC recommender system maps recovery strategy with faults aspects as follows:

1- Retry: this strategy is eligible to handle “Availability Fault” with “Major Severity Level”.
2- Substitution: this strategy is best used to handle “Dependency / integration fault” with “Minor Severity Level”, “Availability Fault” with “Major Severity Level” and “Right-First-Time failure” with “Blocker Level”. However, this strategy is optimum for the stated fault types, but it also can be used to handle other faults.
3- Roll-back: this strategy is eligible to handle “Dependency / integration fault” with “Minor Severity Level” and “Test path delay fault” with “Low Severity Level”.
4- Replication: this strategy is eligible to handle “Availability Fault” with “Major Severity Level”.
5- Checkpointing: this strategy is eligible to handle “Control Flow error” with “Critical Severity Level”.

Table 2 represents each fault type and severity level with its optimum corresponding recovery strategy determined by F2RS-WSC recommender system.

Table 2 F2RS-WSC recovery strategy Mapping

Recovery Strategy	Fault Type	Severity Level
Retry	Availability Fault	Major Severity Level
Substitution	Dependency / integration fault, Right-First-Time failure, and Availability Fault	Minor Severity Level, Blocker Level and Major Severity Level
Roll-Back	Test path delay fault and Dependency / integration fault	Low Severity Level and Minor Severity Level
Replication	Availability Fault	Major Severity Level
Checkpointing	Control Flow error	Critical Severity Level

4. The Experimental evaluation and results

An overview of the F2RS-WSC approach experimentation results and evaluation metrics is presented in this section. Thus, evaluation is conducted in terms of time metric as well as performance evaluation metrics (precision, recall, f-measure, and accuracy). F2RS-WSC is developed using python on PyCharm platform. The experimentation was carried out on an Intel(R) Core (TM) i-8550U, 2GHz processor and 16GB RAM. To ensure accuracy and efficiency of results, experiments were carried out multiple times on various datasets. The following sub-sections describe in detail the used dataset, the evaluation metrics, and the associated achieved results.

4.1.Datasets

This subsection provides a detailed description of the datasets that were utilized for evaluating the F2RS-WSC approach. The description includes the size, origin, and brief information about each dataset. The "Services Dependency Graphs for Web Services Composition Modeling Dataset" [32] was used, which consists of multiple SDGs generated by the MISD model [5] for four public services datasets. The MISD model was employed to address the challenges faced by other SDGs, while also maintaining the accuracy of the testing process and quality of composition modeling. The layered graph structure of the MISD model simplifies the search process, with each layer containing a selection of services from the repository based on their actual dependencies and WSMI-based metric. The WSMI-based metric assesses actual service dependencies by analyzing various criteria such as input/output parameter matching, parameter count, and the total number of edges utilized by each web service, regardless of any user request. In the F2RS-WSC approach, the WSMI-based SDG is used to create test paths for textual services. The dataset sizes are provided, including the number of services (nodes of SDG) and the number of edges included in each SDG within the dataset file as follows:

1. The dataset described in [2] is Gabrel's web service repository consisting of nine services with input and output parameters. It also includes the SDG generated by the MISD model used as an input for the F2RS-WSC approach.
2. The dataset brought forward by [33] is the Pablo Rodríguez Mier Dataset with information on five service repositories ranging from 1,000 to 9,000 web services. Details for each repository include the original web services with input and output parameters, a generated repository with random parameters, and generated SDGs using the MISD model, which serves as the input for the F2RS-WSC approach.
3. The WSC08 dataset is for the web service challenge held in 2008 [34]. It comprises eight service repositories with original web services, input/output parameters, ranging from 158 to 8,119 services per repository, and the generated SDGs by the MISD model, serving as the input for the F2RS-WSC approach.
4. The WSC09 dataset is for the web service challenge held in 2009 [35]. It consists of five service repositories containing original web services with their corresponding input and output parameters, varying between 527 and 15,211 services per repository, and the generated SDGs by the MISD model used as the input for the F2RS-WSC approach.

4.2.Time Metrics

In this sub-section time metric is evaluated through conducting experiments on F2RS-WSC approach among all datasets described previously in section 4.1. F2RS-WSC used this time metric to measure the recovery strategy recommendation time, in addition to the SDG parsing time and the paths validation time. Thus, the recovery strategy recommendation time represents the time needed by F2RS-WSC approach to recommend the optimum recovery strategy per fault. While the SDG parsing time represents the time consumed by F2RS-WSC approach to parse the given SDG and store all its data in a database table. In addition, for the path validation time it measures the amount of time required to validate a composition and determines if it is faulty or non-faulty after inspecting all its edges and services. Moreover, F2RS-WSC performed a comparison between the 3-time measurements.

The experiments were conducted on all datasets and results were analyzed as shown in Figure 3 and Figure 4. Thus, it was observed that the recovery strategy recommendation time is very minor in comparison with the SDG parsing time and path validation time. Hence, parsing time is the highest among the 3-time measurements as it represents the time consumed by F2RS-WSC to parse SDG and store its data. Accordingly, it is directly proportional to the size of repository thus, as the number of services in a given repository increases the parsing time increases as well. However, from the advantages of F2RS-WSC that the SDG parsing process is only performed once and can be used an infinite number of times for validating infinite paths as well as recommending recovery strategy for infinite number of faults. Meanwhile, the SDG parsing time is much higher than that of the composition/path validation time with about 93% approximately. Thus, the path validation time is a search process on the parsed SDG generated by the MISD model, which is a very well-organized layered graph based on the WSMI value [5].

As for the recovery strategy recommendation time, it is much lower than that of the SDG parsing time by an average of 97.7%. While the recovery strategy recommendation time represents an average of 4% of the path validation time, which makes it lower than that of the path validation time by an average of 96%. Figure 3 presents the 3-time measurements; SDG parsing time, path validation time and Recovery strategy recommendation time of the proposed F2RS-WSC approach conducted on multiple datasets, where the x-axis represents the datasets, and the y-axis represents the 3-time measurements consumed by F2RS-WSC approach in milliseconds. Figure 4 represents the average of the three-time metrics (SDG parsing time, path validation time and Recovery strategy recommendation time) among all datasets, where the x-axis is the 3-time metrics, while y-axis represents the time consumed in milliseconds.

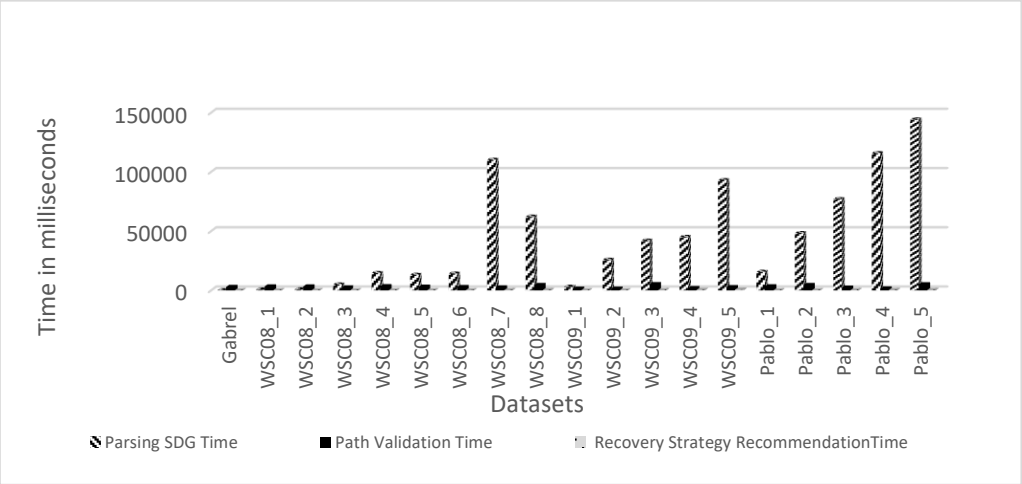


Figure. 3: The F2RS-WSC Time Metrics Evaluation

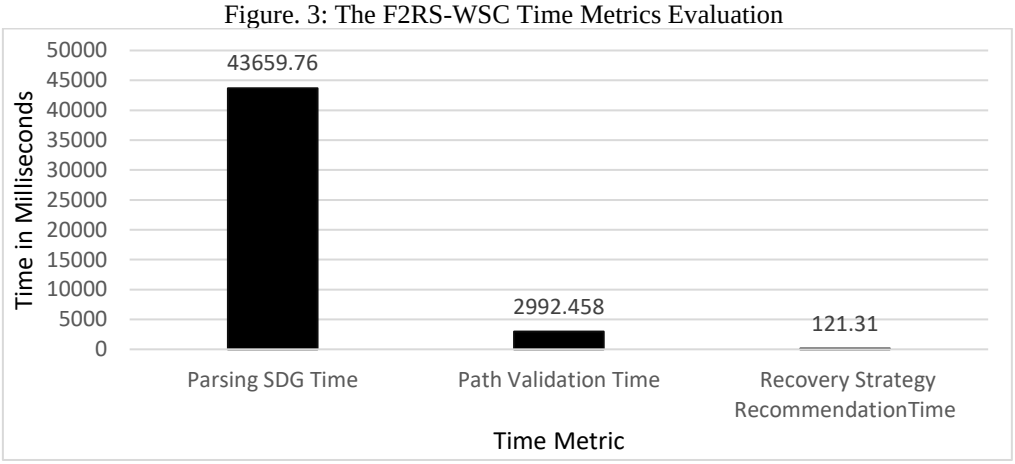


Figure. 4: The average time consumed by the proposed F2RS-WSC approach among all datasets

4.3.Performance Metrics

In this section, the evaluation method used by the F2RS-WSC approach is presented through performance metrics. These metrics are employed to assess the accuracy, reliability, and quality of the F2RS-WSC approach [1]. The performance metrics involve precision, recall, f-measure, and accuracy. Table 3 provides a concise explanation of these metrics along with the formulas that are applied for measuring them based on the confusion matrix, which is presented in Table 4 [36]. The accuracy metric is used to validate F2RS-WSC among the different datasets as per the formula presented in Table 3 based on the Confusion matrix presented in Table 4 [36]. The calculation involves determining the proportion of accurate predictions made in relation to the overall number of predictions [7]. Table 5 represents the results of the F2RS-WSC approach performance metrics among all the datasets.

Table 3. The performance metrics used by F2RS-WSC approach for evaluation

Metric	Meaning	Formula
Accuracy	This illustrates the percentage of accurate predictions in relation to the overall number of predictions made.	$\frac{TP + TN}{TP + FP + TN + FN}$
Recall	This statement denotes the accurate identification of faults that match the actual faults.	$\frac{TP}{TP + FN}$

Precision	It shows the ratio of accurately identified faults to the overall number of classified faults	$\frac{FP}{FP + TN}$
F-measure	It is the harmonic mean of precision and recall	$2 * \frac{(Precision * Recall)}{(Precision + Recall)}$

Table 4. The Confusion matrix

	No (Predicted)	Yes (Predicted)
No (Actual)	True negatives (TN)	False negatives (FP)
Yes (Actual)	False negatives (FN)	True positives (TP)

The F2RS-WSC approach achieves an accuracy level of 70% - 88%. In addition, the correlation between the size of the dataset and the measurement of accuracy is directly proportional; as the dataset size increases the accuracy level increases. Figure.5 shows accuracy measurement of F2RS-WSC approach among all datasets.

Table 5 The Performance metrics for the F2RS-WSC approach

Dataset	Accuracy	Recall	Precision	F-measure
Gabrel Example	70 %	0.7	0.81	0.84
WSC_08_1	75.01 %	0.75	0.89	0.85
WSC_08_2	75.5 %	0.75	0.88	0.88
WSC_08_3	77.6 %	0.77	0.82	0.81
WSC_08_4	79 %	0.79	0.87	0.82
WSC_08_5	82.6 %	0.82	0.89	0.86
WSC_08_6	80.7 %	0.80	0.88	0.89
WSC_08_7	83.2 %	0.83	0.83	0.82
WSC_08_8	84.7%	0.84	0.86	0.87
WSC_09_1	75 %	0.75	0.88	0.88
WSC_09_2	86 %	0.86	0.85	0.81
WSC_09_3	86.4 %	0.86	0.80	0.87
WSC_09_4	87 %	0.87	0.82	0.81
WSC_09_5	88 %	0.88	0.9	0.83
Pablo_1	82.7 %	0.82	0.88	0.89
Pablo_2	83.1 %	0.83	0.81	0.83
Pablo_3	83.9 %	0.83	0.87	0.86
Pablo_4	87.4 %	0.87	0.88	0.88
Pablo_5	87.8 %	0.88	0.85	0.99

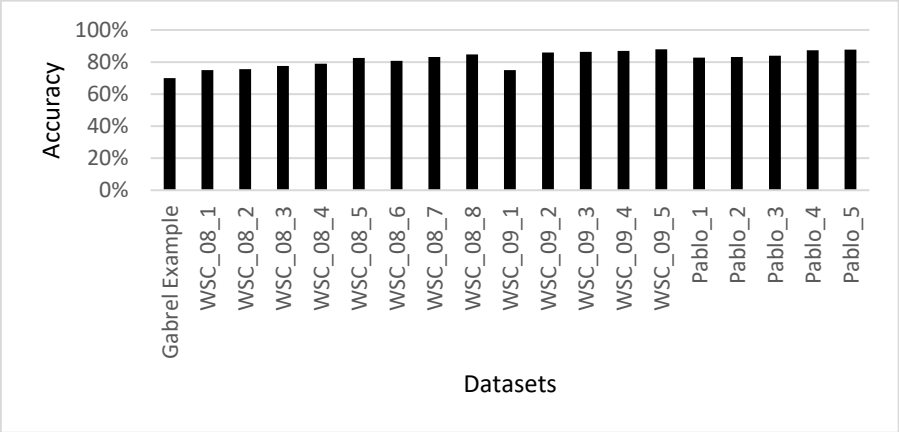


Figure. 5: The accuracy measurement of F2RS-WSC approach among all datasets

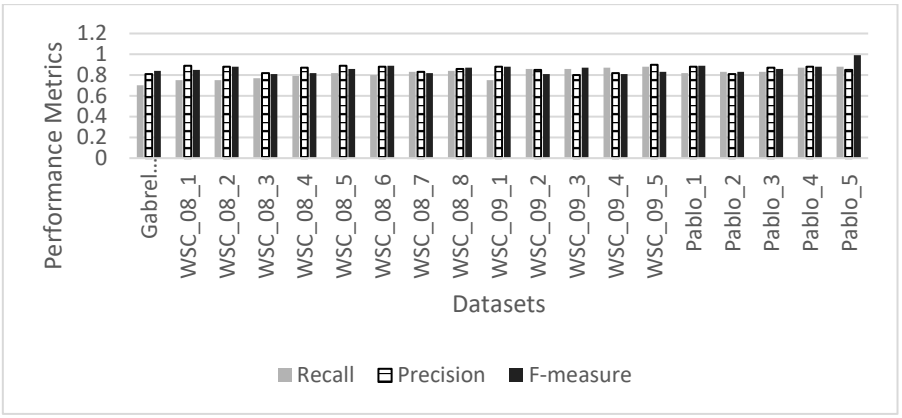


Figure. 6: The performance metric values of F2RS-WSC approach among all datasets

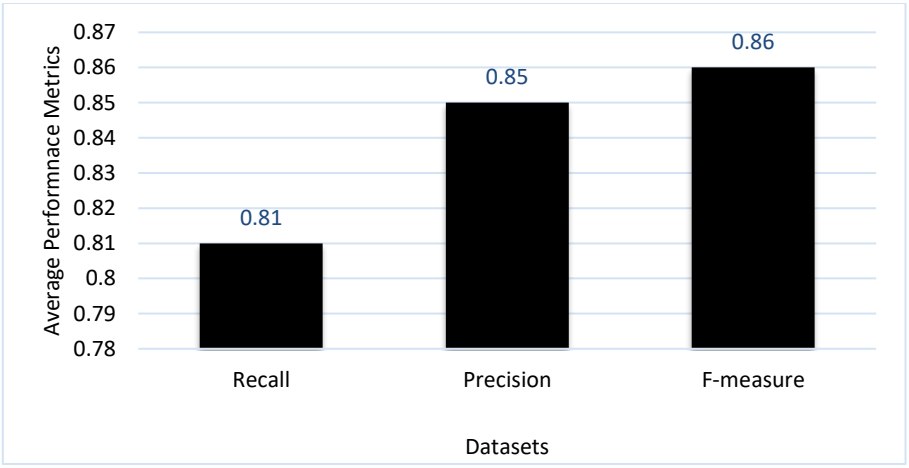


Figure. 7: The average of performance metrics of F2RS-WSC approach among all datasets

While other metrics such as Precision, Recall, and F-measure are commonly used to assess the quality of software [7]. The Precision metric determines the percentage of accurately identified outcomes, whereas Recall evaluates the proportion of correctly identified outcomes out of all possible relevant outcomes. Furthermore, the F-measure combines both Precision and Recall into a unified measurement that captures these attributes [7]. For precision, recall and f-measures the F2RS-WSC approach achieves

averages of 0.85, 0.81 and 0.86 respectively. Thus, Figure 6 represents performance metrics results of the F2RS-WSC approach for all the dataset files, whereas Figure 7 presents the average performance measurements among all datasets.

5. Discussion

This section discusses the advantages and unresolved challenges of utilizing the proposed approach F2RS-WSC. The outcomes from assessing the experimental metrics and comparing them with existing approaches indicate the numerous benefits of using the F2RS-WSC approach. However, there are still some outstanding issues that we need to address, which will be discussed below. As per the conducted experiments and results, the proposed F2RS-WSC approach:

1. Is a model-based approach. Thus, being language-independent makes it more homogeneous with web service compositions, due to the opaque nature with hidden source code.
2. Is a recommender system for fault recovery able to recommend the optimum recovery strategy suitable for the existing fault based on multiple aspects of the fault as fault type, fault severity level, fault's location and time at which fault occurs.
3. Accepts any SDG rather than simple directed graph, as an input as in most of the related studies, to maintain the integration and dependency considerations.
4. Is inspected on multiple datasets of different sizes to assure its scalability, via various evaluation metrics to evaluate its efficiency and effectiveness.

As an on-going process of continuous development, we are working on the following open issues to include in our proposed F2RS-WSC as follows. F2RS-WSC approach should be extended to:

1. Develop a model for handling the tackled faults based on the recommended optimum recovery strategies.
2. Examine multiple models, such as trees, petri nets…etc., and compare their performance to SDGs.
3. Address more fault perspectives, such as introducing faults deliberately through injection.

6. Conclusion

The burgeoning in service-oriented computing (SOC) paradigms lead the way for more convoluted systems. Hence, as the complexity of compositions increases and integration issues arise while combining services, the occurrence of faults tends to rise. Therefore, there is a crucial need to develop remedies for fault recovery to uphold the reliability and resilience of web service compositions. However, the majority of existing studies tend to use substitution recovery strategy to recover faulty services in a given composition. In addition, others apply random recovery strategies without taking into consideration the occurred fault aspects as fault`s type, severity level, fault`s location…etc. In this paper, the Fault Recovery Strategies Recommender System for web service compositions (F2RS-WSC) is proposed to recommend the best recovery strategy for handling emerging faults in the web service compositions paradigm. The proposed system is a model-based system that recommends the best strategy for recovering faulty paths generated from service dependency graphs based on the faults` types, severity levels, faults` location, as well as the time at which faults may occur. The model-based approaches suit the blind nature of web service compositions with hidden source code.

The experimental results obtained to evaluate the proposed F2RS-WSC approach on a variant set of datasets with different sizes considered two main evaluation metrics: time and performance. The results indicate that the time consumed for recovery strategy recommendation in the proposed F2RS-WSC represents less than 3% of the time needed to parse the input SDG. While it represents 4% of path validation time. The path validation time is very minor compared to the parsing time as it represents an average of 6% of the SDG parsing time since the layered SDG minimizes the search-space that directly affects the time. Regarding the performance metric, the applied metrics were accuracy, precision, recall and f-measure. The F2RS-WSC shows an average accuracy between 70% and 88%, in which the accuracy level increases as the size of dataset increases. In addition, the precision, recall and f-measure also gave promising results of averages 0.85,0.81 and 0.86 respectively. Thus, performance metrics indicated that the proposed F2RS-WSC approach overall performance is high which assures the accuracy and efficiency of recommendations. As future work, we plan to develop a comprehensive fault handling model including all fault perspective as fault injection, fault prediction, fault localization and fault recovery. In addition, we plan to compare performance when using different models, i.e., petri-nets, trees...etc. as an input to F2RS-WSC to boost its performance.

References

[1] R. Elghondakly, S. Moussa, N. Badr, Handling Faults in Service Oriented Computing : A Comprehensive Study, Int. Conf. Comput. Sci. Its Appl. Springer, Cham, 2020. (2020) 947–959. doi:10.1007/978-3-030-58811-3.

[2] V. Gabrel, M. Manouvrier, K. Moreau, C. Murat, QoS-aware automatic syntactic service composition problem: Complexity and resolution, Futur. Gener. Comput. Syst. 80 (2018) 311–321. doi:10.1016/j.future.2017.04.009.

[3] S.A. Raza Kazmi, A. Qasim, A. Khalid, R. Assad, M. Shahbaz, Formal modeling and verification of cloud-based web service composition, Concurr. Comput. Pract. Exp. (2019) e5249. doi:10.1002/cpe.5249.

[4] A. Barkat, Framework for web service composition based on QoS in the multi cloud environment, Int. J. Inf. Technol. (2021). doi:10.1007/s41870-020-00564-z.

[5] R.A. Elghondakly, S.M. Moussa, N.L. Badr, Mutual Information-based Modeling for Services Dependency, IEEE Trans. Serv. Comput. (2022) 1–18. doi:10.1109/TSC.2022.3207232.

[6] C. Jatoth, G.R. Gangadharan, U. Fiore, R. Buyya, QoS-aware Big service composition using MapReduce based evolutionary algorithm with guided mutation, Futur. Gener. Comput. Syst. 86 (2018) 1008–1018. doi:10.1016/j.future.2017.07.042.

[7] R. Elghondakly, S.M. Moussa, N. Badr, Service-oriented model-based fault prediction and localization for service compositions testing using deep learning techniques, Appl. Soft Comput. 143 (2023) 110430. doi:10.1016/j.asoc.2023.110430.

[8] R.A. Elghondakly, S.M. Moussa, N.L. Badr, The DSW Model: An Efficient Approach for Single Web Services Modeling, 2021 Tenth Int. Conf. Intell. Comput. Inf. Syst. (ICICIS), 2021. (2021) 500–505. doi:10.1109/ICICIS52592.2021.9694204.

[9] K.S.M. Chan, J. Bishop, J. Steyn, L. Baresi, A Fault Taxonomy for Web Service Composition, Int. Conf. Serv. Comput. Springer, Berlin, Heidelberg, 2009. (2009). doi:https://doi.org/10.1007/978-3-540-93851-4_36.

[10] X. Chen, J.-H. Jiang, A method of virtual machine placement for faulttolerant cloud applications, Intell. Autom. Soft Comput. 22 (2016) 587–597.

[11] L. Liu, J. Shang, Fault Tolerance for Web Service based on Component Importance in Service Networks, Proc. Fifth Int. Conf. Network, Commun. Comput. (Pp. 103-109). (2016) 103–109. doi:10.1145/3033288.3033328.
[12] J. Liu, J. Zhou, Software Rejuvenation based Fault Tolerance Scheme for Cloud Applications, 2015 IEEE 8th Int. Conf. Cloud Comput. (Pp. 1115-1118). IEEE. (2015) 1115–1118. doi:10.1109/CLOUD.2015.164.
[13] M. Vargas-Santiago, S.E.P. Hernández, L.A. Morales-Rosales, H.H. Kacem, Survey on Web Services Fault Tolerance Approaches Based on Checkpointing Mechanisms, J. Softw., 12(7), 507-525. 12 (2017) 507–525. doi:10.17706/jsw.12.7.507-525.
[14] Y. Chen, L., Fan, G., & Liu, A formal method to model and analyse QoS-aware fault tolerant service composition, Int. J. Comput. Sci. Eng. 12 (2016).
[15] R. Gupta, R. Kamal, U. Suman, A QoS-supported approach using fault detection and tolerance for achieving reliability in dynamic orchestration of web services, Int. J. Inf. Technol. (2017). doi:10.1007/s41870-017-0066-z.
[16] H. Fekih, S. Mtibaa, S. Bouamama, The dynamic approach for fault-tolerance service reconfiguration composition based on multi-level VCSOP web service composition based a on multi-level VCSOP, Procedia Comput. Sci. 159 (2019) 1527–1536. doi:10.1016/j.procs.2019.09.323.
[17] A. Kaur, G. Kaur, Resource scheduling based on load balancing with fault tolerance in cloud computing, Int. Adv. Res. J. Sci. Eng. Technol.,. 5 (2018) 59–66.
[18] Thakur, R., K. Verma, Fault-tolerant web service composition using a hybrid approach of checkpointing and recommender system, Int. J. Grid High Perform. Comput. 10 (2018) 13–27.
[19] Reddy, C. V., A. P Shukla, A critique of recommender systems in web service composition., 2017 Fourth Int. Conf. Parallel, Distrib. Grid Comput. , IEEE. (2017) 336–341.
[20] A.A. Adebiyi, L.S. Aro, A.A. Onashoga, S. A., Akinsanya, O.J. Oyelade, Hybrid state space planning and recommendation system for web service composition in cloud environment, Futur. Gener. Comput. Syst. 91 (2019) 59–72.
[21] P. Veeresh, R.P. Sam, C.S. Bindu, Reliable fault tolerance system for service composition in mobile Ad Hoc network, Int. J. Electr. Comput. Eng. Vol. 9 (2019) 2523–2533. doi:10.11591/ijece.v9i4.pp2523-2533.
[22] R. Angarita, M. Rukoz, Y. Cardinale, Modeling dynamic recovery strategy for composite web services execution, World Wide Web, Springer. (2016) 89–109. doi:10.1007/s11280-015-0329-1.
[23] T. Laleh, J. Paquet, S. Mokhov, Y. Yan, Constraint verification failure recovery in web service composition, Futur. Gener. Comput. Syst. 89 (2018) 387–401. doi:10.1016/j.future.2018.06.037.
[24] G.P. Bhandari, R. Gupta, Extended Fault Taxonomy of SOA-Based Systems, 25 (2017) 237–257. doi:10.20532/cit.2017.1003569.
[25] S. Kumar, D.S. Rana, S.C. Dimri, Fault Tolerance and Load Balancing algorithm in Cloud Computing : A survey, Int. J. Adv. Res. Comput. Commun. Eng. 4(7), 92-96. (2015) 92–96. doi:10.17148/IJARCCE.2015.4720.
[26] R. Angarita, M. Rukoz, Reliable Composite Web Services Execution : Towards a Dynamic Recovery Decision, Electron. Notes Theor. Comput. Sci. 302 (2014) 5–28. doi:10.1016/j.entcs.2014.01.018.
[27] D. Pandey, R. Shankar, R. Pathak, An adaptive approach for dynamic recovery decisions in web service composition using space based QoS factor., Int. J. Web Serv. Comput. 6 (2015).
[28] S.L. Fan, Y. Bin Yang, X.X. Wang, Efficient web service composition via knapsack-variant algorithm, Lect. Notes Comput. Sci. Artif. Intell. Bioinformatics). 10969 LNCS (2018) 51–66.

doi:10.1007/978-3-319-94376-3_4.

[29] P. Rodriguez-Mier, M. Mucientes, M. Lama, Automatic web service composition with a heuristic-based search algorithm, Proc. - 2011 IEEE 9th Int. Conf. Web Serv. ICWS 2011. (2011) 81–88. doi:10.1109/ICWS.2011.89.

[30] R.S. Chhillar, Empirical Analysis of Object-oriented Design Metrics for Predicting High , Medium and Low Severity Faults using Mallows CP, ACM SIGSOFT Softw. Eng. Notes. 36 (2011) 1–9. doi:10.1145/2047414.2047423.

[31] T. Yanagisawa, Y. Tamura, A. Anand, S. Yamada, Comparison of Hazard-Rates Considering Fault Severity Levels and Imperfect Debugging for OSS, J. Softw. Eng. Appl. (2021) 591–606. doi:10.4236/jsea.2021.1411035.

[32] R. [dataset] ElGhondakly, S. Moussa, Services Dependency Graphs for Web Services Composition Modeling Dataset, IEEE Dataport, V1. (2022). doi:https://dx.doi.org/10.21227/xacr-xh31.

[33] P. Rodriguez-Mier, M. Mucientes, M. Lama, Hybrid optimization algorithm for large-scale QoS-aware service composition, IEEE Trans. Serv. Comput. (2015) 1–17. doi:10.1109/TSC.2015.2480396.

[34] A. Bansal, M.B. Blake, T. Weise, M.C. Jaeger, D.-B. Germany, WSC-08 : Continuing the Web Services Challenge, 2008 10th IEEE Conf. E-Commerce Technol. Fifth IEEE Conf. Enterp. Comput. E-Commerce E-Services. IEEE. (2008) 351–354. doi:10.1109/CEC/EEE.2008.67.

[35] S. Kona, A. Bansal, M.B. Blake, S. Bleul, T. Weise, WSC-2009: A quality of service-oriented web services challenge, 2009 IEEE Conf. Commer. Enterp. Comput. CEC 2009. (2009) 487–490. doi:10.1109/CEC.2009.80.

[36] G. Bhandari, R. Gupta, S.K. Upadhyay, An approach for fault prediction in SOA-based systems using machine learning techniques, Data Technol. Appl. 53(4), 397-421. (2019). doi:10.1108/DTA-03-2019-0040.